THE UNIVERSITY OF
MELBOURNE

COMP 90048
Declarative Programming
Workshop 7 (week8)

2019 semester 1

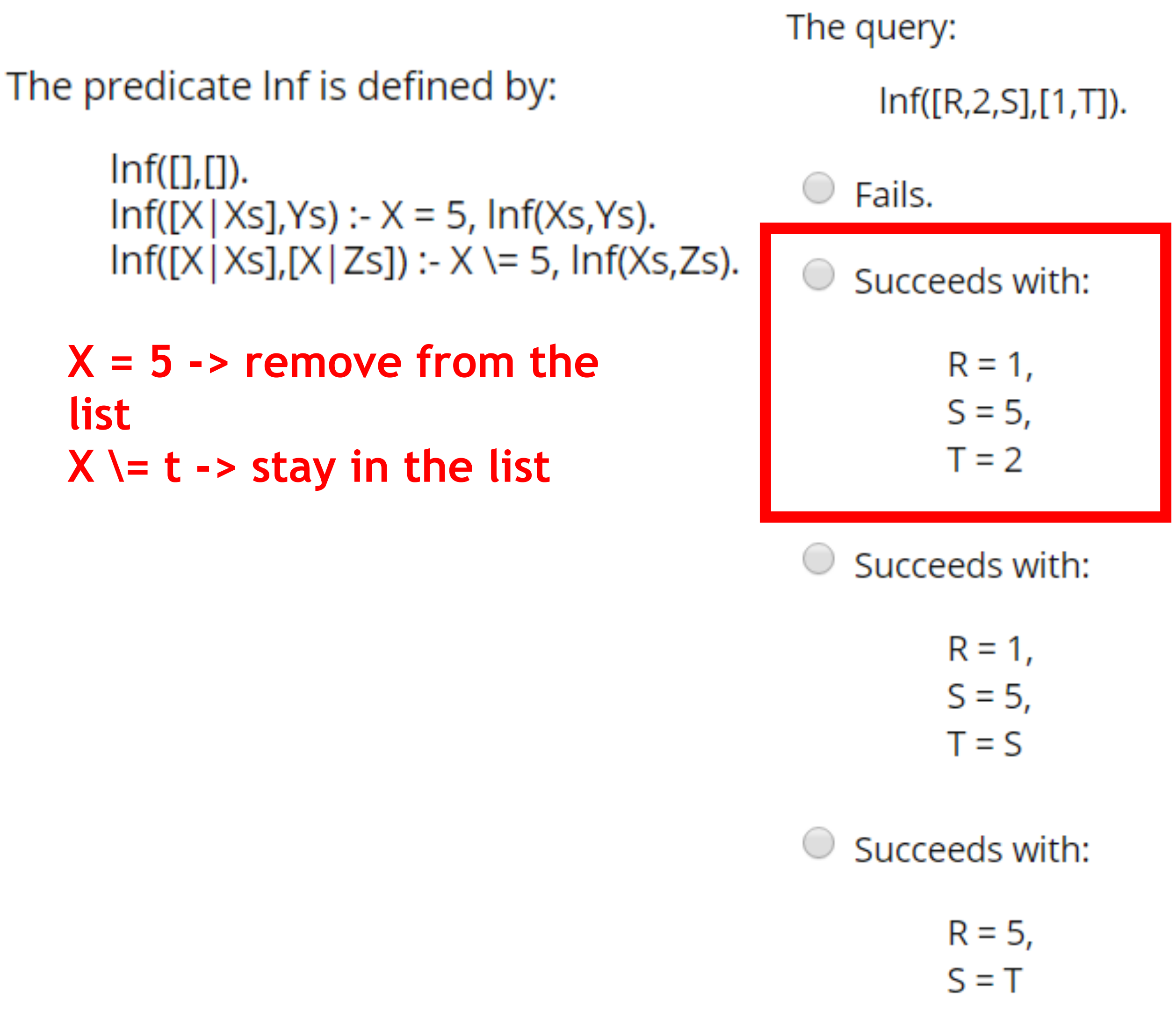
by Wendy Zeng

Tutorial : Tue 18:15 - 19:15 221 Bouverie St, room B11
Wed 17:15 - 18:15 201 Bouverie St, room B132

Outline

1. Prolog List
   1. Member
   2. Append
2. Prolog Tree
   1. Tree member
   2. Tree insert
3. Term inspection

Warming Up



The predicate Inf is defined by:

```
Inf([],[]).
Inf([X|Xs],Ys) :- X = 5, Inf(Xs,Ys).
Inf([X|Xs],[X|Zs]) :- X \= 5, Inf(Xs,Zs).
```

The query:

```
Inf([R,2,S],[1,T]).
```

○ Fails.

○ Succeeds with:

R = 1,
S = 5,
T = 2

○ Succeeds with:

R = 1,
S = 5,
T = S

○ Succeeds with:

R = 5,
S = T

X = 5 -> remove from the list
X \= t -> stay in the list

1. Prolog List

- Haskell List: Homogeneous (x:xs)
  - All elements are of same type, a Haskell list is of type [t] as a list of some type t
- Prolog List: Heterogeneous [Head|Tail]
  - Elements can be of different types
  - Example: X = [a, 1, 2.5, [c, d]].

1. member:

```
member(E, [E|_]).
member(E, [_|R]) :- member(E, R).
```

1. Prolog List

- 2. Prolog append:

```
append([], Back, Rest).
append([H|T], Back, [H|R]) :-
    append(T, Back, R).
```

- Haskell append:

```
(++) [] back = back
(++) (x:xs) back =
    x : (++) xs back
```

- What we can do with append:

  - Member: append(_, [E|_], L).
  - Last element: append(_, [E], L).
  - Adjacent elements: append(_, [E1, E2|_], L).
  - Remove prefix: append([a,b], X, [a,b,1,2,3]).
  - Remove suffix: append(Y, [1,2,3], [a,b,1,2,3]).
  - Split a list into two parts: append(X, Y, [a,b,1,2,3]).

1. Prolog List

- 3. Reverse:

①
```
reverse1([], []).
reverse1([Head|Tail], Rev) :-
            reverse1(Tail, Rev_tail),
            append(Rev_tail, [Head], Rev).
```

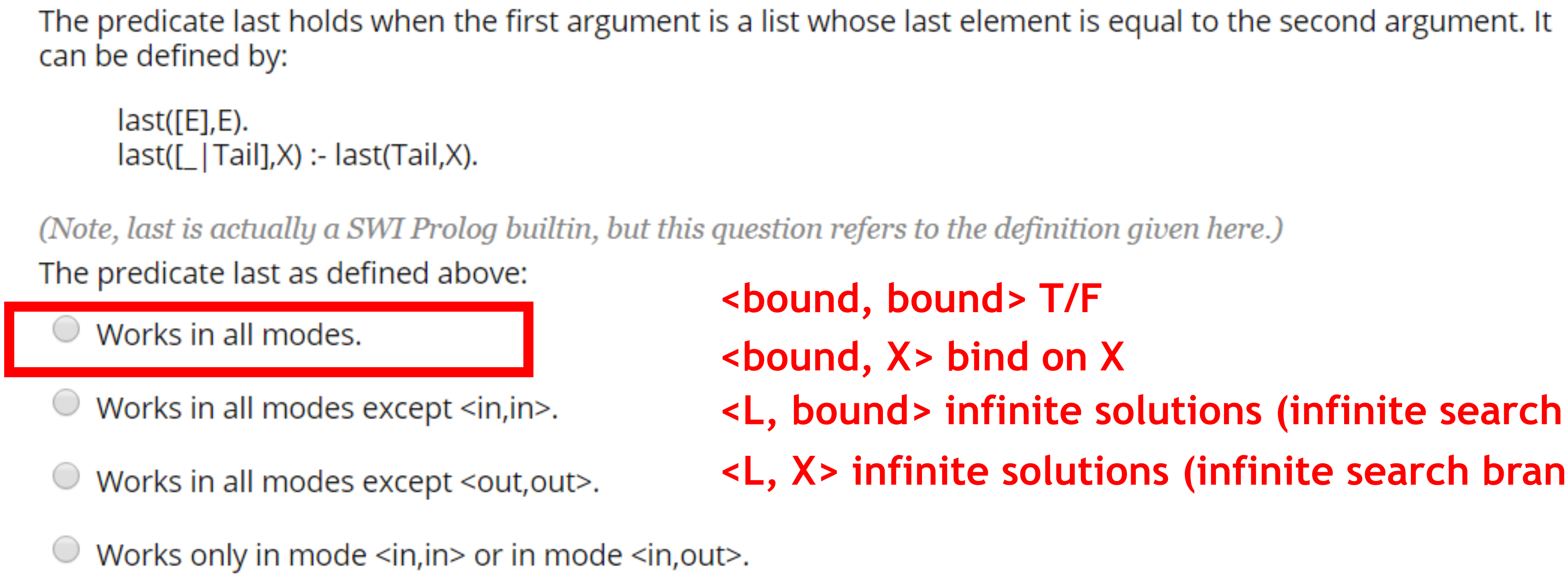
②
```
reverse2([], []).
reverse2([Head|Tail], Rev) :-
            append(Rev_tail, [Head], Rev),
            reverse2(Tail, Rev_tail).
```

Infinite backtracking loop (exhaustive enumeration of solutions)

③
```
reverse3(ABC, CBA) :-
            samelength(ABC, CBA),
            reverse1(ABC, CBA).

samelength([], []).
samelength([_|Xs], [_|Ys]) :- samelength(Xs, Ys).
```

3. Prolog List



The predicate last holds when the first argument is a list whose last element is equal to the second argument. It can be defined by:

```
last([E],E).
last([_|Tail],X) :- last(Tail,X).
```

(Note, last is actually a SWI Prolog builtin, but this question refers to the definition given here.)

The predicate last as defined above:

[✓ ○ Works in all modes.]

○ Works in all modes except <in,in>.

○ Works in all modes except <out,out>.

○ Works only in mode <in,in> or in mode <in,out>.

<bound, bound> T/F

<bound, X> bind on X

<L, bound> infinite solutions (infinite search branch)

<L, X> infinite solutions (infinite search branch)

2. Prolog Tree

Representation:

Atom: leaf

Compound term: tree(L, V, R)

1. Tree member:

intset_member(+N, +Set).

```
intset_member(N, tree(_,N,_)).
intset_member(N, tree(L,N0,_)) :-
    N < N0, intset_member(N, L).
intset_member(N, tree(_,N0,R)) :-
    N > N0, intset_member1(N, R).
```

2. Prolog Tree

2. Tree insert: Insert a value into an existing tree

intset_insert(+N, +Set0, -Set)

```
intset_insert(N, leaf, tree(leaf, N, leaf)).
intset_insert(N, tree(L, N, R), tree(L, N, R)).
intset_insert(N, tree(L, N0, R), tree(L1, N0, R)) :-
    intset_insert(N, L, L1), N<N0.
intset_insert(N, tree(L, N0, R), tree(L, N0, R1)) :-
    intset_insert(N, R, R1), N>N0.
```

Set0: tree with 0 or 1 occurrences of value N

Set1: tree with exactly 1 occurrences of value N

3. Term inspection

1. Compound term

   a. functor:

      1). Decompose: functor(+CompoundTerm, -FunctorName, -Arity)
      2). Create compound term:functor(-CompoundTerm, +FunctorName, +Arity)

   b. arg:

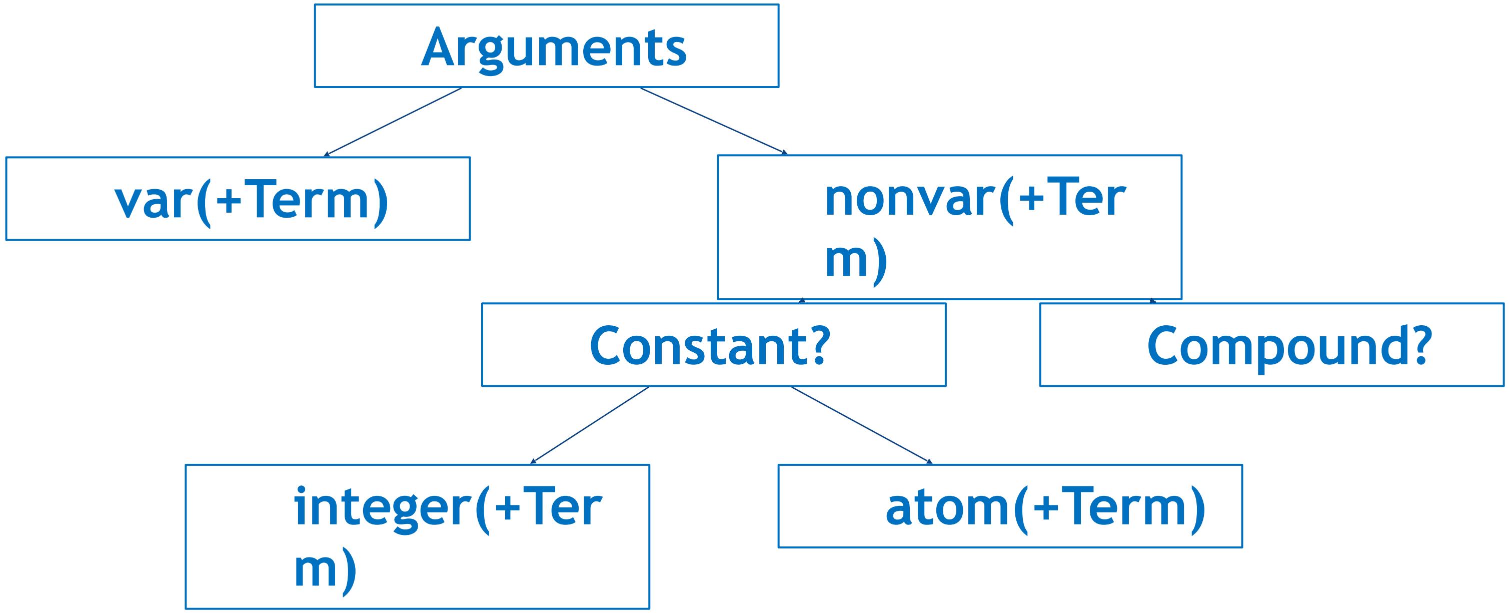
      1). Access a particular argument at a certain position:
         arg(+ArgIndex, +CompoundTerm, -Arg)
      2). Instantiate a variable for particular argument:
         arg(+ArgIndex, -CompoundTerm, +Arg)

   c. =..

      Decompose compound term into a list of
      [FunctorName, Arg1, Arg2 ….]

3. Term inspection

2. Groundness: ground(+Term)
3. Prolog type inspection:



THE UNIVERSITY OF
MELBOURNE

Thank you

wendy.zeng@unimelb.edu.au

By Wendy Zeng